

Symbolic Analysis and Parameter Synthesis for **Parametric Timed Automata** and **Petri Nets** using **Maude with SMT Solving**

Peter Ölveczky (University of Oslo)

Based on joint work with **Carlos Olarte**, **Jaime Arias**, **Laure Petrucci** (Sorbonne Paris-Nord); **Kyungmin Bae** (POSTECH); and **Fredrik Rømming** (Cambridge)

IFIP WG 1.3 Meeting, Salzburg, Feb 5–8, 2024

Background

Goal

Maude-with-SMT for Parametric Timed Automata

Parametric Interval Time Petri Nets with Inhibitor Arcs (PITPNs)

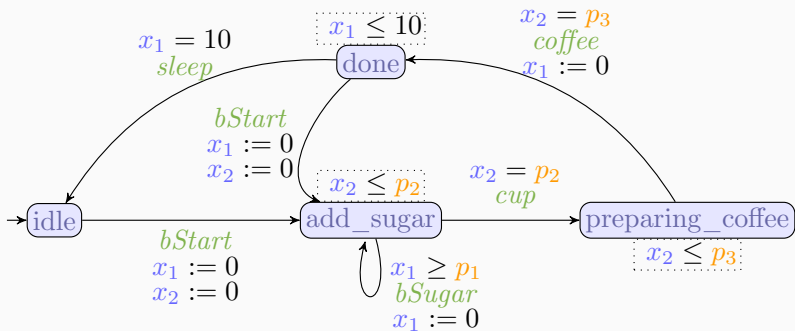
Networks of Parametric Timed Automata with Variables

Concluding Remarks

Background

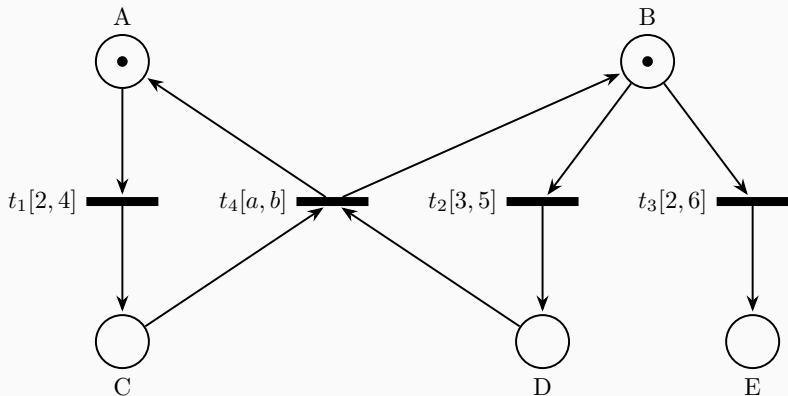
Context: Formal Models for Real-Time Systems

- Timed automata



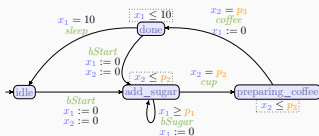
Context: Formal Models for Real-Time Systems

- Timed automata
- Time(d) Petri nets

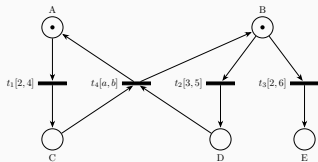


Context: Formal Models for Real-Time Systems

- Timed automata



- Time(d) Petri nets



- **Limited** expressiveness/modeling convenience

- data types; unbounded data structures; fixed communication;

...

- + many properties **decidable**

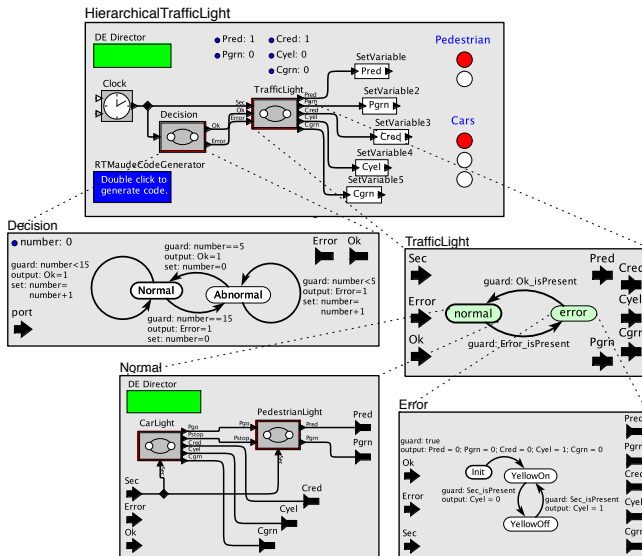
- Expressiveness and modeling convenience
 - algebraic specification!

- Expressiveness and modeling convenience
 - algebraic specification!
- User-defined data types; flexible communication; unbounded data structures; dynamic object/message creation/deletion; hierarchical states; ...

- Expressiveness and modeling convenience
 - algebraic specification!
- User-defined data types; flexible communication; unbounded data structures; dynamic object/message creation/deletion; hierarchical states; ...
- Found unknown flaws in
 - scheduling with reuse of unused budgets (unbounded queues)
 - wireless sensor algorithms (communication, areas, angles)
 - MANET protocol (mobility + communication delay)
 - 50-page multicast protocol (large functions; communication)
 - cloud-based transaction systems
 - cars in Japan
 - avionics systems (complex dynamics; LTL model checking)
 - ...

Semantics and Analysis for Modeling Languages

Ptolemy II DE Models:



Semantics and Analysis for Modeling Languages

Ptolemy II DE Models:

codeDirectory:

generatorPackage:

generateComment:

inline:

overwriteFiles:

run:

Simulation bound:

Safety Property:

Alternation Property:

Error Handling:

Code Generator Commands

Check
mc-tctl {init} |= AG (('HierarchicalTrafficLight . 'Decision))(port 'Error is present)implies AF[<= than 1] 'HierarchicalTrafficLight |('Cyel = # 1,'Cgrn = # 0,'Cred = # 0)) .
in PTOLEMY-MODELCHECK with mode maximal time increase

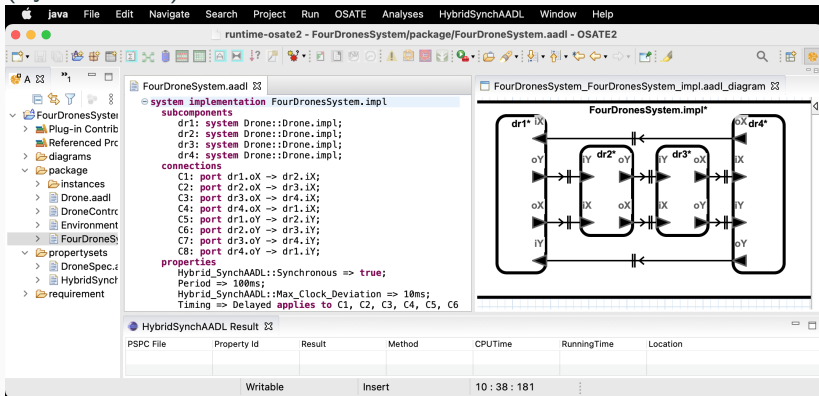
Checking equivalent property:
mc-tctl {init} |= not (E tt U[>= than 0] ('HierarchicalTrafficLight . 'Decision))(port 'Error is present) and (E not 'HierarchicalTrafficLight |('Cgrn = # 0,'Cred = # 0,'Cyel = # 1)U[> than 1] tt) .

Property not satisfied

All Done

Semantics and Analysis for Modeling Languages

(Synchronous) AADL:



The screenshot displays the OSATE IDE interface for editing an AADL file. The main editor shows the implementation of the `FourDronesSystem` package, which consists of four drone components connected in a ring topology.

```
FourDronesSystem.impl
system implementation
  subcomponents
    dr1: system Drone::Drone.impl;
    dr2: system Drone::Drone.impl;
    dr3: system Drone::Drone.impl;
    dr4: system Drone::Drone.impl;
  connections
    C1: port dr1.oX -> dr2.iX;
    C2: port dr2.oX -> dr3.iX;
    C3: port dr3.oX -> dr4.iX;
    C4: port dr4.oX -> dr1.iX;
    C5: port dr1.oY -> dr2.iY;
    C6: port dr2.oY -> dr3.iY;
    C7: port dr3.oY -> dr4.iY;
    C8: port dr4.oY -> dr1.iY;
  properties
    Hybrid_SynchAADL::Synchronous => true;
    Period => 100ms;
    Hybrid_SynchAADL::Max_Clock_Deviation => 10ms;
    Timing => Delayed applies to C1, C2, C3, C4, C5, C6
```

The diagram on the right, titled `FourDronesSystem.impl*`, visualizes this implementation. It shows four drone components (`dr1*`, `dr2*`, `dr3*`, and `dr4*`) arranged in a square. Each component has two output ports (`oX`, `oY`) and two input ports (`iX`, `iY`). The connections are as follows: `dr1*` outputs to `dr2*` (C1), `dr2*` outputs to `dr3*` (C2), `dr3*` outputs to `dr4*` (C3), and `dr4*` outputs to `dr1*` (C4). Additionally, there are cross-connections between adjacent drones: `dr1*` to `dr2*` (C5), `dr2*` to `dr3*` (C6), `dr3*` to `dr4*` (C7), and `dr4*` to `dr1*` (C8).

At the bottom of the IDE, a `HybridSynchAADL Result` table is visible:

PSPC File	Property Id	Result	Method	CPUTime	RunningTime	Location
	Writable		Insert	10 : 38 : 181		

Semantics and Analysis for Modeling Languages

(Synchronous) AADL:

The screenshot displays the OSATE IDE interface. The main editor shows the AADL code for `FourDronesSystem.aadl` and its implementation `FourDronesSystem.impl`. The code includes subcomponents, connections, and properties. A menu is open over the `Formal Analysis` option, showing sub-options like `Symbolic Reachability`, `Randomized Simulation`, and `Portfolio Analysis`. The `reachability` analysis results are visible in the right-hand pane, showing an invariant and two propositions.

```
FourDronesSystem.aadl
@system implementation FourDronesSystem.impl
  subcomponents
    dr1: system Drone::Drone.impl;
    dr2: system Drone::Drone.impl;
    dr3: system Drone::Drone.impl;
    dr4: system Drone::Drone.impl;
  connections
    C1: port dr1.oX -> dr2.iX;
    C2: port dr2.oX -> dr3.iX;
    C3: port dr3.oX -> dr4.iX;
    C4: port dr4.oX -> dr1.iX;
    C5: port dr1.oY -> dr2.iY;
    C6: port dr2.oY -> dr3.iY;
    C7: port dr3.oY -> dr4.iY;
    C8: port dr4.oY -> dr1.iY;
  properties
    Hybrid_SynchAADL::Synchronous => true;
    Period => 100ms;
    Hybrid_SynchAADL::Max_Clock_Deviation => 10ms;
    Timing => Delayed applies to C1, C2, C3, C4, C5, C6
```

```
reachability
?initial ==> true;

@invariant [safety] :
?initial ==> not(?collision) in time 500;

@proposition [initial] :
abs(dr1.env.x - 1.1) < 0.01 and abs(dr1.env.y - 1.5) <
abs(dr2.env.x + 1.5) < 0.01 and abs(dr2.env.y + 1.1) <
abs(dr3.env.x - 1.5) < 0.01 and abs(dr3.env.y - 1.1) <
abs(dr4.env.x + 1.1) < 0.01 and abs(dr4.env.y + 1.5) <

@proposition [gather] :
abs(dr1.env.x - dr2.env.x) < 1.0 and abs(dr1.env.y - d
abs(dr1.env.x - dr3.env.x) < 1.0 and abs(dr1.env.y - d
abs(dr1.env.x - dr4.env.x) < 1.0 and abs(dr1.env.y - d
abs(dr2.env.x - dr3.env.x) < 1.0 and abs(dr2.env.y - d
abs(dr2.env.x - dr4.env.x) < 1.0 and abs(dr2.env.y - d
abs(dr3.env.x - dr4.env.x) < 1.0 and abs(dr3.env.y - d
```

PSPC File	Property Id	Result	Method	CPUTime	RunningTime	Location
	Writable	Insert		9 : 9 : 201		

Rewriting Logic

- static parts: algebraic equational specification
- dynamic parts: rewrite rules

`cr1 [l] : t => u if cond .`

Rewriting Logic

- **static parts**: algebraic equational specification
- **dynamic parts**: rewrite rules

`cr1 [l] : t => u if cond .`

Maude: Language and Analysis for Rewriting Logic

- distributed systems in OO style
- simulation
- (explicit-state) search, LTL model checking, ...
- meta-programming
- unification and narrowing

Recent connection to SMT solvers: Yices2, CVC4, Z3

```
check t .
```

```
smt-search t =>* u such that ... . --- undocumented!
```

```
op metaCheck : Module Term ~> Bool [special (...)] .  
    --- meta-level
```


Maude with SMT Solving “in Practice”

- **Topmost rules** (grab whole state)
- Symbolic “terms/states”

$$\Phi(x, y, \dots) \parallel t(x, y, \dots)$$

Maude with SMT Solving “in Practice”

- **Topmost rules** (grab whole state)
- Symbolic “terms/states”

$$\Phi(x, y, \dots) \parallel t(x, y, \dots)$$

- Rule $l : q \longrightarrow r$ **if** ψ turned into rule

$$l : \text{PHI} \parallel q \longrightarrow (\text{PHI and } \psi) \parallel r \text{ **if** } \text{smtCheck}(\text{PHI and } \psi)$$

Real-Time Rewrite Theories

- equational specification for data types
- rewrite rules for **instantaneous** change
- **tick rules**

`cr1 [/] : {t} => {u} in time τ if ...`

model **time advance**

Real-Time Rewrite Theories

- equational specification for data types
- rewrite rules for instantaneous change
- tick rules

`crl [/] : {t} => {u} in time τ if ...`

model time advance

Real-Time Maude

- explicit-state simulation, reachability analysis, ...
- unbounded and time-bounded analysis
- timed CTL model checking

- Tick rules often

```
cr1 [tick] : {t} => {f(t,x)} in time x if x <= mte(t) .
```

- x new variable
- non-executable

Explicit-State Analysis in Real-Time Maude

- Tick rules often

```
cr1 [tick] : {t} => {f(t,x)} in time x if x <= mte(t) .
```

- x new variable
- non-executable
- Real-Time Maude: explicit-state analysis

Explicit-State Analysis in Real-Time Maude

- Tick rules often

```
cr1 [tick] : {t} => {f(t,x)} in time x if x <= mte(t) .
```

- x new variable
- non-executable
- Real-Time Maude: explicit-state analysis
- Real-Time Maude: sample certain moments in time
 - all times not visited
 - analysis unsound/incomplete
 - “counterexamples” real

Explicit-State Analysis in Real-Time Maude

- Tick rules often

cr1 [*tick*] : $\{t\} \Rightarrow \{f(t, x)\}$ in time x if $x \leq mte(t)$.

- x new variable
- non-executable
- Real-Time Maude: explicit-state analysis
- Real-Time Maude: sample certain moments in time
 - all times not visited
 - analysis unsound/incomplete
 - “counterexamples” real

Real-Time Maude: expressive, but “unsound”

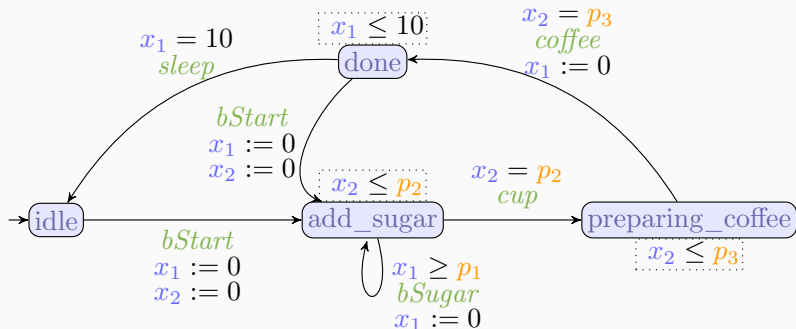
Goal

Research Goals

- Sound and complete analysis methods for systems beyond timed automata, ...
- Efficient rewriting-with-SMT methods for real-time systems
- Time increase encoded symbolically

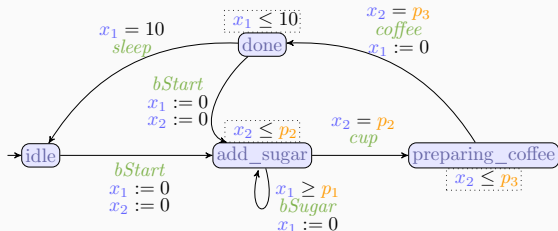
Maude-with-SMT for Parametric Timed Automata

Parametric Timed Automata (PTA)



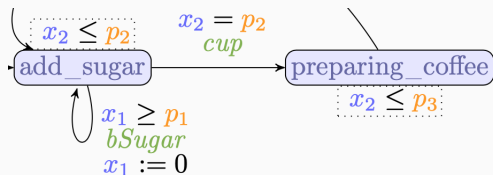
Imitator: state-of-the-art PTA parameter synthesis tool (É. André)

Representing PTA in Maude



```
mod PTA-COFFEE is pr REAL .  
  sorts State NState DState Location .  
  subsorts NState DState < State .  
  vars X1 X2 P1 P2 P3 T : Real .  
  
  ops idle addSugar preparingCoffee done : -> Location .  
  
  op <_:_;_> <_;;_> : Location Real Real Real Real Real -> DState .  
  
  op [_:_;_] <_;;_> : Location Real ... -> NState .
```

Representing PTA in Maude (cont.)



```
cr1 [addSugar-cup] :
  < addSugar : X1 ; X2 >          < P1 ; P2 ; P3 >
=>
  [ preparingCoffee : X1 ; X2 ] < P1 ; P2 ; P3 >
if (X2 == P2 and X2 <= P3) = true .
```

```
cr1 [preparingCoffee-tick] :
  [ preparingCoffee : X1 ; X2 ]          < P1 ; P2 ; P3 >
=>
  < preparingCoffee : X1 + T ; X2 + T > < P1 ; P2 ; P3 >
  if (X2 + T <= P3 and T >= 0/1) = true [nonexec] .
```

Analysis with Maude's smt-search

Is there a **parameter valuation**, from initial state with same clock values, such that we reach state **done**?

```
smt-search [ idle : X1 ; X2 ] < P1 ; P2 ; P3 > =>*  
          < done : X1' ; X2' > < P1 ; P2 ; P3 >  
  such that (X1 == X2 and X1 >= 0/1 and P1 >= 0/1  
            and P2 >= 0/1 and P3 >= 0/1) = true .
```

Analysis with Maude's smt-search

Is there a **parameter valuation**, from initial state with same clock values, such that we reach state **done**?

```
smt-search [ idle : X1 ; X2 ] < P1 ; P2 ; P3 > =>*
           < done : X1' ; X2' > < P1 ; P2 ; P3 >
           such that (X1 === X2 and X1 >= 0/1 and P1 >= 0/1
                       and P2 >= 0/1 and P3 >= 0/1) = true .
```

Solution 1

```
state: < done : #3-T ; #1-T + #2-T + #3-T > <P1 ; P2 ; P3>
```

```
where X1 === X2 and X1 >= 0/1 and P1 >= 0/1 and P2 >= 0/1 and
... and #1-T:Real + #2-T:Real === P3 and
... and #3-T:Real <= 10/1 and #1-T:Real + #2-T:Real === P3
```

At least $X2' \geq P3$...

Maude's `smt-search`

- stops exploring when **same** symbolic state seen
- tick application adds fresh variable to state
 - search for unreachable states does not terminate

Reachability

Maude's smt-search

- stops exploring when **same** symbolic state seen
- tick application adds fresh variable to state
 - search for unreachable states does not terminate

Our reachability Command

- stop when symbolic state **subsumed by** earlier symbolic state
- $\phi_u \parallel t_u \sqsubseteq \phi_v \parallel t_v$
iff exists σ such that $t_u = t_v\sigma$ and $\phi_u \implies \phi_v\sigma$

Reachability

Maude's smt-search

- stops exploring when **same** symbolic state seen
- tick application adds fresh variable to state
 - search for unreachable states does not terminate

Our reachability Command

- stop when symbolic state **subsumed by** earlier symbolic state
- $\phi_u \parallel t_u \sqsubseteq \phi_v \parallel t_v$
iff exists σ such that $t_u = t_v\sigma$ and $\phi_u \implies \phi_v\sigma$
- defined subsumption-based **reachability** command
 - Maude's **meta-level** (metaMatch and metaCheck)

Reachability

Maude's smt-search

- stops exploring when **same** symbolic state seen
- tick application adds fresh variable to state
 - search for unreachable states does not terminate

Our reachability Command

- stop when symbolic state **subsumed by** earlier symbolic state
- $\phi_u \parallel t_u \sqsubseteq \phi_v \parallel t_v$
iff exists σ such that $t_u = t_v\sigma$ and $\phi_u \implies \phi_v\sigma$
- defined subsumption-based **reachability** command
 - Maude's **meta-level** (metaMatch and metaCheck)

Theorem

reachability terminates when **parametric zone graph** of PTA finite

- + *soundness/completeness results*

- $EF\phi$ -synthesis and $AG\neg\phi$ -synthesis
- ϕ involves locations and clocks
- parameters: $\exists X.\varphi$ of resulting constraint φ
 - quantifier elimination (Z3)
 - done reachable when $p_2 \leq p_3$

Analysis and Parameter Synthesis

- $EF\phi$ -synthesis and $AG\neg\phi$ -synthesis
- ϕ involves locations **and clocks**
- parameters: $\exists X.\varphi$ of resulting constraint φ
 - quantifier elimination (Z3)
 - **done** reachable when $p_2 \leq p_3$
- Analysis with Maude strategies

Example

Can we reach **done** when **some** sugar required? (If $p_1 \leq p_2 \leq p_3$)

Compared to Imitator

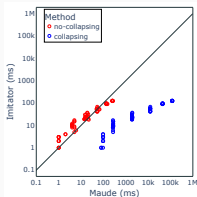
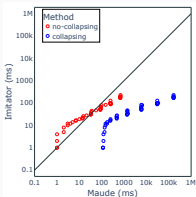
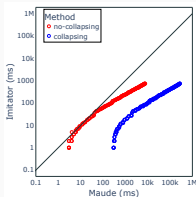
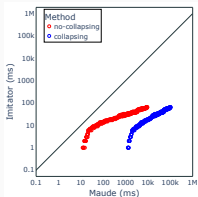
- Terminates when Imitator (without heuristics) terminates
- Do most things Imitator can do
 - not robustness and liveness

Compared to Imitator

- Terminates when Imitator (without heuristics) terminates
- Do most things Imitator can do
 - not robustness and liveness
- More than Imitator
 - analysis/parameter synthesis with **execution strategies**
 - **clocks** and **parameters** in propositions

Compared to Imitator

- Terminates when Imitator (without heuristics) terminates
- Do most things Imitator can do
 - not robustness and liveness
- More than Imitator
 - analysis/parameter synthesis with **execution strategies**
 - **clocks** and **parameters** in propositions
- Benchmarking on PTA library

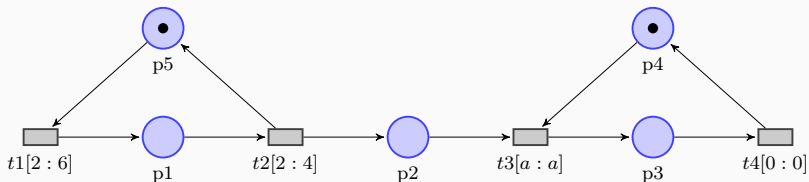


Unfair comparison!

- Promising
- New `reachability` method
- Parameter synthesis
- Very limited model
 - atomic states
 - no equations
 - linear constraints

Parametric Interval Time Petri Nets with Inhibitor Arcs (PITPNs)

Parametric Interval Time Petri Nets with Inhibitor Arcs (PITPNs)



- Unbounded states
- “User-defined” functions
- “Interpreter”

Representing Nets in Maude

Markings represented $p \mapsto 2 ; q \mapsto 3$ instead of $p p q q q$

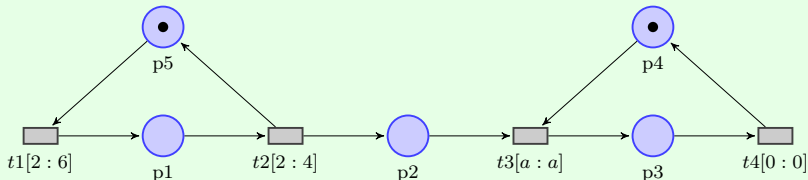
- PITPNs: **interleaving** semantics (!)
- enables **parametric markings** with SMT

Representing Nets in Maude

Markings represented $p \mid \rightarrow 2 ; q \mid \rightarrow 3$ instead of $p p q q q$

- PITPNs: interleaving semantics (!)
- enables parametric markings with SMT

Example



"t1" : "p5" $\mid \rightarrow 1$ \dashrightarrow "p1" $\mid \rightarrow 1$ in [2 : 6] ;

"t2" : "p1" $\mid \rightarrow 1$ \dashrightarrow "p2" $\mid \rightarrow 1$; "p5" $\mid \rightarrow 1$ in [2 : 4] ;

"t3" : "p2" $\mid \rightarrow 1$; "p4" $\mid \rightarrow 1$ \dashrightarrow "p3" $\mid \rightarrow 1$ in [a : a] ;

"t4" : "p3" $\mid \rightarrow 1$ \dashrightarrow "p4" $\mid \rightarrow 1$ in [0 : 0]

Symbolic Rewrite Semantics: Advance Time

```
var T : Real .
```

```
cr1 [tick] : tickOk      : M : CLOCKS      : NET  
            => tickNotOk : M : increaseClocks(M, CLOCKS, NET, T) : NET  
if (T >= 0/1 and mte(M, CLOCKS, NET, T)) = true .
```

```
op mte : Marking ClockValues Net Real -> Boolean .
```

```
eq mte(M, empty, NET, T) = true .
```

```
eq mte(M, (L -> R1) ; CLOCKS, (L : PRE --> ... in [T1 : inf]) ; NET, T)  
= mte(M, CLOCKS, NET, T) .
```

```
eq mte(M, (L -> R1) ; CLOCKS, (L : PRE --> ... in [T1 : T2]) ; NET, T)  
= (active(M, L : ...) ? T <= T2 - R1 : true)  
and mte(M, CLOCKS, NET, T) .
```

Symbolic Rewrite Semantics: Applying Transition

```
cr1 [applyTransition] :
  TS : M : ((L -> T) ; CLOCKS) :
  (L : PRE --> POST inhibit INH in INTERVAL) ; NET)
=>
  tickOk : ((M - PRE) + POST) : updateClocks(.., M - PRE, ....) :
  (L : PRE --> ... ; NET)
if (active(...)) and (T in INTERVAL)) = true .

eq updateClocks((L' -> R1) ; CLOCKS, INTERM-M, (L' : PRE --> ...); NET)
= (L -> PRE <= INTERM-M ? R1 : 0/1) ; updateClocks(...)
```


1-Unsafe Nets Reachable?

```
smt-search  tickOk : m : initClocks(net) : net
            =>* TICK : M : CLOCKS : NET
            such that (a:Real >= 0/1 and not k-safe(1, M)) = true .
```

Unsafe net reachable if $a \geq 4$

- Subsumption $T \sqsubseteq U$ not sufficient
 - $\llbracket T \rrbracket \subseteq \llbracket U \rrbracket$ does **not** imply $T \sqsubseteq U$

- Subsumption $T \sqsubseteq U$ not sufficient
 - $\llbracket T \rrbracket \subseteq \llbracket U \rrbracket$ does **not** imply $T \sqsubseteq U$
- Defined **new subsumption** \preceq
 - $T \preceq U$ iff $\llbracket T \rrbracket \subseteq \llbracket U \rrbracket$ for PITPNs

- Subsumption $T \sqsubseteq U$ not sufficient
 - $\llbracket T \rrbracket \subseteq \llbracket U \rrbracket$ does **not** imply $T \sqsubseteq U$
- Defined **new subsumption** \preceq
 - $T \preceq U$ iff $\llbracket T \rrbracket \subseteq \llbracket U \rrbracket$ for PITPNs

Theorem

Reachability with \preceq -folding terminates when symbolic state space finite!

Reachability Analyses with New Folding

1. **Maude search**: carry symbolic states **visited in current behavior**
 - stop when current state subsumed by previous symbolic state **in same branch**
2. Store **all** visited symbolic states + implement search analysis
 - encode breadth-first search at Maude meta-level

Reachability Analyses with New Folding

1. Maude search: carry symbolic states **visited in current behavior**
 - stop when current state subsumed by previous symbolic state **in same branch**
2. Store **all** visited symbolic states + implement search analysis
 - encode breadth-first search at Maude meta-level

Example

```
search init(net, m0, a < 4 ...)  
  =>* S :  $\phi'$  || ( TICK : M : CLOCKS : NET )  
      such that smtCheck( $\phi'$  and not k-safe(1,M)) .
```

returns No solution

For what parameter values is the net not 1-safe?

```
search [1] init(net, m0,  $\phi$ )  
=>* S : PHI' || ( TICK : M : CLOCKS : NET )  
such that smtCheck(PHI' and not k-safe(1,M)) .
```

For what parameter values is the net not 1-safe?

```
search [1] init(net, m0,  $\phi$ )  
=>* S : PHI' || ( TICK : M : CLOCKS : NET )  
such that smtCheck(PHI' and not k-safe(1,M)) .
```

- quantifier elimination of resulting constraint gives desired values of parameters
- $a \geq 4$

m_s parametric initial marking; $0 \leq x_i \leq 1$ tokens in place p_i

`safety-syn`(*net*, m_s , ϕ_0 , k-safe(1,M))

- net 1-safe when $x_1 = x_3 = 0$ and $0 \leq x_2 \leq 1$

- Analysis with user-defined strategies
- Non-nested temporal properties
- ...

Comparison with Roméo

- Terminates when Roméo terminates
 - **not** vice versa

Comparison with Roméo

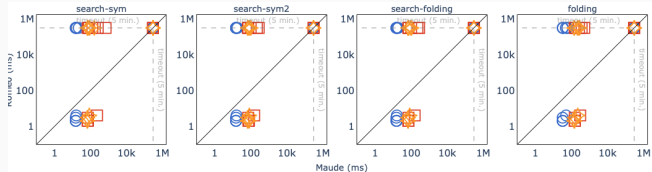
- Terminates when Roméo terminates
 - **not** vice versa
- Do almost everything Roméo can do
 - **not** parameters in temporal property time bounds

Comparison with Roméo

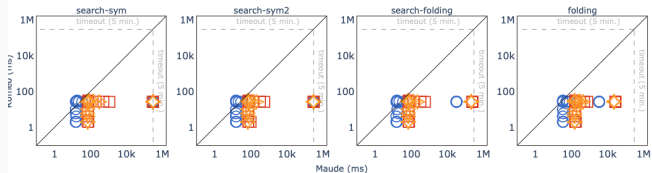
- Terminates when Roméo terminates
 - **not** vice versa
- Do almost everything Roméo can do
 - **not** parameters in temporal property time bounds
- More:
 - parametric/synthesize initial markings
 - user-defined strategies (restricting possible behaviors)
 - ...

Benchmarking

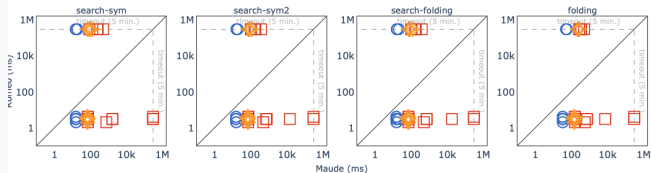
Comparison with Roméo; different implementations and solvers



(a) producer-consumer

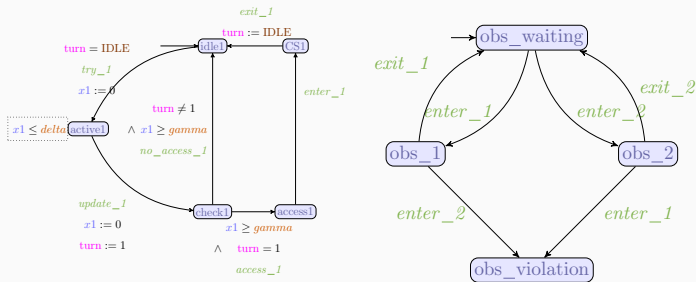


(b) scheduling



Networks of Parametric Timed Automata with Variables

Networks of Parametric Timed Automata with Variables (NPTAVs)



“Interpreter” for NPTAVs

- sound/correct
- folding-based analysis
- terminates whenever PZG finite

Example

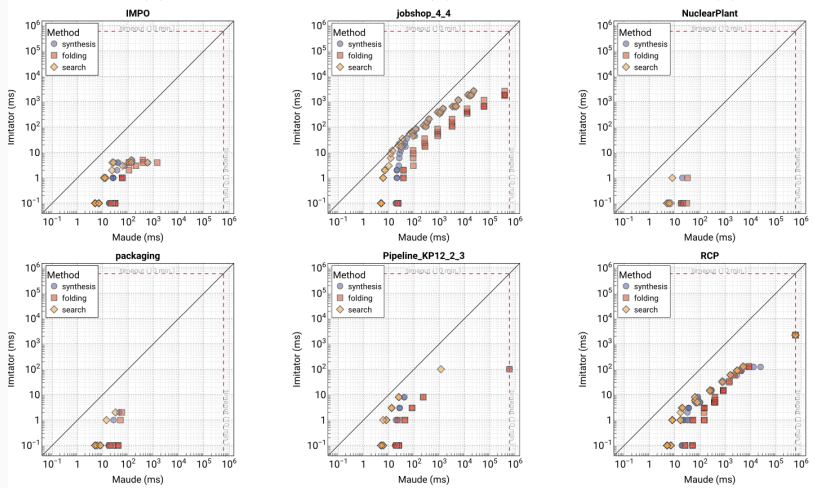
```
Maude> red synthesis-folding in 'FISCHER :  
      init => (observer @ obs-violation) .
```

Example

```
Maude> red synthesis-folding in 'FISCHER :  
        init => (observer @ obs-violation) .  
  
result ConjRelLinRExpr: rr(var(gamma)) + -1 * rr(var(delta)) <= 0  
        and -1 * rr(var(delta)) <= 0 and -1 * rr(var(gamma)) <= 0
```

Bad states reachable when $\gamma \leq \delta$

Benchmarking: Imitator vs Maude-SMT



Concluding Remarks

Concluding Remarks I

- Semantics/interpreter for NPTAVs and PITPNs
- Symbolic states (Maude-with-SMT analysis)
- **New folding**: termination when Imitator/Roméo terminates
- Reachability, parameter synthesis, ...

Concluding Remarks I

- Semantics/interpreter for NPTAVs and PITPNs
- Symbolic states (Maude-with-SMT analysis)
- **New folding**: termination when Imitator/Roméo terminates
- Reachability, parameter synthesis, ...
- Can do essentially everything Imitator/Roméo can do, and ...
 - analysis with **strategies**
 - parametric markings
 - LTL model checking
 - clocks and parameters in **“propositions”**

Concluding Remarks I

- Semantics/interpreter for NPTAVs and PITPNs
- Symbolic states (Maude-with-SMT analysis)
- **New folding**: termination when Imitator/Roméo terminates
- Reachability, parameter synthesis, ...
- Can do essentially everything Imitator/Roméo can do, and ...
 - analysis with **strategies**
 - parametric markings
 - LTL model checking
 - clocks and parameters in “**propositions**”
- Benchmarking
 - worse than Imitator
 - comparable with Roméo (?)

Concluding Remarks II

- Methods towards “Symbolic Real-Time Maude”
- Maude-with-SMT for not-entirely-trivial systems
- Different **subsumption** relations; different reachability analysis methods; Fourier-Motzkin quantifier elimination; parameter synthesis; ...
- Extend to **classes of systems beyond** automata and Petri nets
 - linear arithmetic/reals?
 - topmost rules?
 - ...
- Symbolic full (timed) temporal logic model checking